Sample Midterm Answers

1. What are all possible outputs of the following code fragment?

```
void f(int a, int b)
{
      printf("%d %d\n", a, b);
}

void main(void)
{
      int i = 5;
      f(++i, ++i);
}
```

   Answer: Thekey pointis that the function arguments can be evaluated in any order. So, the function can be called as `f(6, 7)` or `f(7, 6)`. So, the two possible outputs are:

```
6 7
```

   and

```
7 6
```

2. Given the definitions

```
      int numbs[10];
      int *ptr = numbs;
```

   which of the following are equivalent, and why?

   a. numbs[3]
   b. numbs + 3
   c. * (numbs + 3)
   d. * (ptr + 3)
   e. *ptr + 3

   Answer: A refers to the third element of the array numbs. B refers to the address of the third element of the array C refers to the quantity at the address of the third element of the array, which is the third element of the array. As ptr s assigned numb, D refers to the same thing as C. E is the value of the element stored at numb, plus 3. Hence a, c, and d are equivalent.

3. Write a recursive function to add the integers from a to b. You may assume that a ≤ b initially.

   Answer:

```
int add(int a, int b)
{
      /* base case: a == b */
      if (a == b)
            return(b);
      /* add lowest element to sum of rest */
      return(a + add(a+1, b));
}
```

4. Use the following code fragment to answer parts a, b, and c:

```
            for(x = i = 0; i <= 100; i += 2, x += i);
```

   a. In one short sentence , what does this for loop do?

      Answer: It stores in `x` the sum of the even numbers from 0 to 100 inclusive.

   b. Is the following while loop equivalent? If not, how does its result differ?

```
      x = i = 0;
      while( i++ = 100)
```

```
        x += ++i;
```

Answer: No. The loop has a syntax error, because `i++` cannot be assigned to. If the expression in the while condition were `i++ <= 100`, though, then the program will compile. However, the loop places in `x` sums the even numbers from 0 to 102, because when i is 100, `i++ <= 100` is true (remember, the value of `i` is used before the "++" operator increments `i`).

c. Does the following for loop do the same thing? If not, what does it do?

```
    for(x = i = 0; i <= 100; i++){
          if (!(i % 2))
                continue;
          x = x + i;
    }
```

Answer: No. This sums the odd integers from 0 to 100 inclusive and stores the value in `x`.

5. What does the following function do?

```
    int x(char *s, char *t)
    {
          for( ; *s == *t; s++, t++)
                if (*s == '\0')
                      return(0);
          return(*s – *t);
    }
```

Answer: It retuns 0 if the two argument strings are the same, and the difference between the first characters in which the argument strings differ otherwise. This is (essentially) the *strcmp*(3) function.

6. What does this function do?

```
char *x(char *s, char c)
{
      char *r = NULL;

      do{
            while(*s && *s != c) s++;
            if (*s) r = s;
      } while(*s++);
      return(r);
}
```

Answer: It returns a pointer to the last occurrence in argument `s` of the character in argument `c`. If the character does not occur in that string, it returns the NULL pointer.

7. The following segment of code is supposed to print the number of times the routine a_again is called. Yet, regardless of the input, it prints 0. Why? How would you fix it?

```
void a_again(int acount)
{
      ++acount;
}

void main(void)
[
      register int c;
      int counter = 0;

      while((c = getchar()) != EOF)
            if (c == 'a' || c == 'A')
                  a_again(counter);
```

```
        printf("%d\n", counter);
        exit(0);
}
```

Answer: The problem is that acount is passed as a parameter to a_again. As C calls by value, not reference, the value of counter is not changed by a_again. One way to fix this is to make counter global, and not pass anything to a_again:

```
int counter = 0;
void a_again()
{
        ++counter;
}

void main(void)
[
        register int c;

        while((c = getchar()) != EOF)
                if (c == 'a' || c == 'A')
                        a_again();

        printf("%d\n", counter);
        exit(0);
}
```